DEVELOPMENT OF A PARALLEL MESH GENERATOR USING QUADTREES

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

by

S. P. ABHYANKAR

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
September, 1991

ACKNOWLEDGEMENT

I wish to express my sincere gratitude towards my guide Dr. S.G.Dhande for his guidance and invaluable suggestions. I am also thankful to Dr. P.Datta from Mathematics Dept., for allowing me to work on Transputers, Dr. Siddiqui, Harriprasad of ERNET Lab for PAR._C manual, Dr. Shastri Pradhan from CAD Lab, I am also thankful to all my friends and Prasad who have helped me a lot while completing this project.

SUVARNA ABHYANKAR.

ABSTRACT

Fast automatic mesh generation schemes are required in many pplication areas such as finite element analysis. The present ork outlines one such methodology of mesh generation that can be mplemented on a parallel architecture of processors. The proposed esh generation algorithm uses a modified quadtree approach. The uadtree data structure seems to be the natural choice for the arallel architecture that has been used. The proposed approach rovides the robust control over the mesh density. The basic arallel architecture used for implementation is a network of four 800 transputers connected in a star network. The present mplementation generates a fully automatic mesh for two limensional geometries of any shape, singly or multiply connected. he work done is in PAR_C and provides some example results also.

Table of contents

CHAPTER 1

INTRODUCTION

1.1 Background Information

The discretisation of the given geometry is widely studied problem as it has got wide applications. These problems are fundamental to many areas such as solid modeling, computer aided design, graphical rendering and scientific computations.

A partitioning problem of particular interest, is optimal triangualtion of a planar point sets. This problem finds applications in image processing, contour mapping, cartography, finite element analysis.

Finite element analysis is an important area of application. The evolution of finite element method has been one of the most notable advances in engineering practices over last 25 years. Typical applications are in Aerospace, Civil and structural Engg, steel and concrete bridges, soil mechanics and fluid dynamics. Finite element analysis is commonly used to obtain information about structural behaviour without building prototypes. Today it is primarily used as an after the fact verification tool. Using

FEM in the earlier stages of the design can improve the design concept. Actual modelling and prototyping is not necessary.

The following steps are followed for analysing the structure:-

1. A number of attributes such as material properties to define physical behaviour under elastic deformation, loadcase includes applied forces and restraints and thus simulates a test case and future position of the part within the assembly.

2. A mesh is generated from the boundary description of the part. This divides the interior of the objects into simple elements of the type known to the application program. In practice quadrilaterals or triangles for 2D and bricks, wedges and tetrahedrons for 3D are used. The mesh is a discrete approximation of the continuous interior of the object. It is important that the difference between mesh and object should be small. The quality of the appoximation is resolution.

3. The loadcase information is assigned to elements and nodes and the mesh is analysed. The computations include matrix approximations such as inversion or Guass type elimination. The cost of analysis is largely dependent upon the size of stiffness matrix which itself is a function of number of degrees of freedom to which the accuracy is inversely proportional and dependent further on mesh properties. The analysis is the discrete approximation of the continuous deformation of the part. Therefore more or higher order elements are required to model accurately regions of dramatic changes in the strain energy. Element numbering, global node numbering and nodal coordinate

information together with additional element and nodal information such as boundary node data are required for these calculations.

Thus inherent in all FEM methods is the necessity to divide the region of interest in a finite number of points or nodes. This subdivision is termed as mesh generation. This process plays an important role in the analysis of a design and also one of the more sensitive stages in the process as it is intimately linked with both the geometry of the path and particular load case concerned. The accuracy of finte element solution is dependent on the element subdivision employed and generally makes largest contribution to the error involved.

The basic criterion satisfied by any FE mesh subdivision is that piecewise variation of the basic unknown from element to element closely approximates the true solution. Finite element solution performed with increasingly finer mesh division essentially converge into the exact solution of the problem. But it can be computationally expensive in the sense that larger size matrix handling is involved with more number of elements. Thus good balance of accuracy and cost is achieved.

The mesh generation is very time consuming and tedious job,but is the essential part of the process. It can be error prone if done manually.This problem was identified long back and the need of automation was felt almost at the same time as of the automation of FE method itself. Automation of the process eventually relieves the user of tedious task of geometry definition and visualisation of the mesh, which leaves him with the essential work for preparing the model,its attributes and specific loadcases.

Because of the advancement of the various CAD tools, it is possible to support visualisation and verification of the structure of nodes within the elements and of elements within the mesh.

1.2 Objectives and Scope

Faster processors and parallel architectures are becoming widely available. This development has motivated the present work to exploit the possibilities of using such architectures for faster FE methods. Though there are serious efforts towards the parallelisation of the FE method itself, as its inherent nature makes it attractive option for parallelisation, there are very few attempts to parallelise the mesh generation. So this work is the attempt to check such possibility. The advantages of parallel implementation are quite obvious.

The objective of this work is to generate fully automatic mesh generation parallely. It is implemented on the parallel architectural class of *transputers*. The approach used here for the mesh generation is the *modified quadtree approach*.

There are various aprroaches, which are discussed in the next chapter, to the mesh genertion . But the quadtree method is an attractive alternative in view of the parallelisation of the process and seemed to be the natural choice for the architecture we used. The architecture we have used is a 4-node T800 tranputer network connected in star, as there is the minimum communication required amongst the worker nodes.

The system designed gives the valid triangular meshes for most of the

arbritarily shaped two dimensional geometries which are singly or multipaly connected.

Although it has been limited to 2D domains it can easily be extended for 3D domains. Our aim is to provide a system designed to give the fully automated, faster, parallel model for mesh generation which gives the valid triangular meshes for most of the arbritarily shaped two dimensional geometries singly or multipaly connected. The work is limited rather to mesh generation than to the actual Finite Element Method requirements. There are efforts to give the valid meshes as far as possible. But truly speaking the fully automted mesh gneration is not viable because in most of the cases the mesh generation depends upon not only the geometry but there are various factors such as boundary conditions, load distribution etc which really influence the final mesh generation requirements. So features for interactvity should be provided for any practical mesh generators.

1.3 Organisation

The remainder of the thesis is organised as follows :

In chapter 2 various mesh generation techniques, serial as well as parallel, have been discussed and compared. It also contains the detailed discussion about the modified quadtree technique which has been primarily used.

In chapter 3, spatial hierarchial structures like quadtrees and octrees have been discussed. Various algorithms, serial as well as parallel have also been discussed. Its merits and demerits have been checked.

In the last section, the actual implementation details has been provided.

CHAPTER 2

MESH GENERATION

2.1 INTRODUCTION

Fundamental to the automation of FEM is the ability to generate a computational grid. It is the process of generating finite element models for simulated structural analysis.

A fully automatic mesh generator is defined as the one which takes as an input ,a geometric representation and associated mesh control information and automatically generates the valid mesh without any manual intervention. A valid mesh is topologically compatible and geometrically similar to the model. Topological compatibility is a requirement that the mesh topology and geometric topology are congruent , while geometric similarity is a

requirement that the mesh geometry is an acceptable refinement which matches the geometric model exactly. Automation of the process relieves the user of tedious task of geometry definition and visualization of the mesh ,which leaves him or her with the essential task for preparing the model ,its attributes and the specific load cases. Mesh generation plays an important role in the analysis of a design and also one of the more sensitive stages in the process as it is intimately linked with both the geometry of the part and the particular load case concerned.

The accuracy of the FE solution depends on the element mesh layout and the cost of analysis becomes prohibitively expensive if the number of elements in the mesh are too high. A good mesh generating scheme, therefore, should help the user in generating a mesh that is just fine enough for an adequate solution accuracy and efficiency. The conformity of the elements should also be assured to avoid gaps in the structure.

The requirements of a good element are:

1. Element should not be too elongated
2. Areas of steep gradient of stress are expected to have finer elements subdivision
3. Quadrilateral elements do not offer much flexibility and mesh grading is a problem in these cases. So generally the triangular elements are preferred. Also in case of triangular elements compatibility is assured.
4. The triangular elements should not be elongated wherever possible and triangles should not be obtuse

angle triangles as this produces stiffness matrix which may become singular or unstable

Approaches to automatic mesh generation are many and varied substantially and are influenced heavily by the particular geometric modeler. The focus of mesh generation has moved from co-ordinate transformation scheme which is input sensitive and automatic triangulation which is fairly robust but involves time consuming searching to the hierarchical spatial decomposition schemes which are versatile robust and are based on certain principles of computational geometry which enable them to function without applying too many heuristics.[CHE89]

2.2 SERIAL 2D MESH GENERATION

There are serious efforts of automating the mesh generations part. There are over 250 methods and most of which deal with manually partitioning the region with quadrilaterals and occasionally to the triangular zones. Although there are lot of different methods adopted for serial mesh generation very little literature is available on parallel mesh generation. Whatever material is available on parallel mesh generators is from the view of parallel triangulation of given points; the technique which can possibly be used for mesh generation.[CHE89]

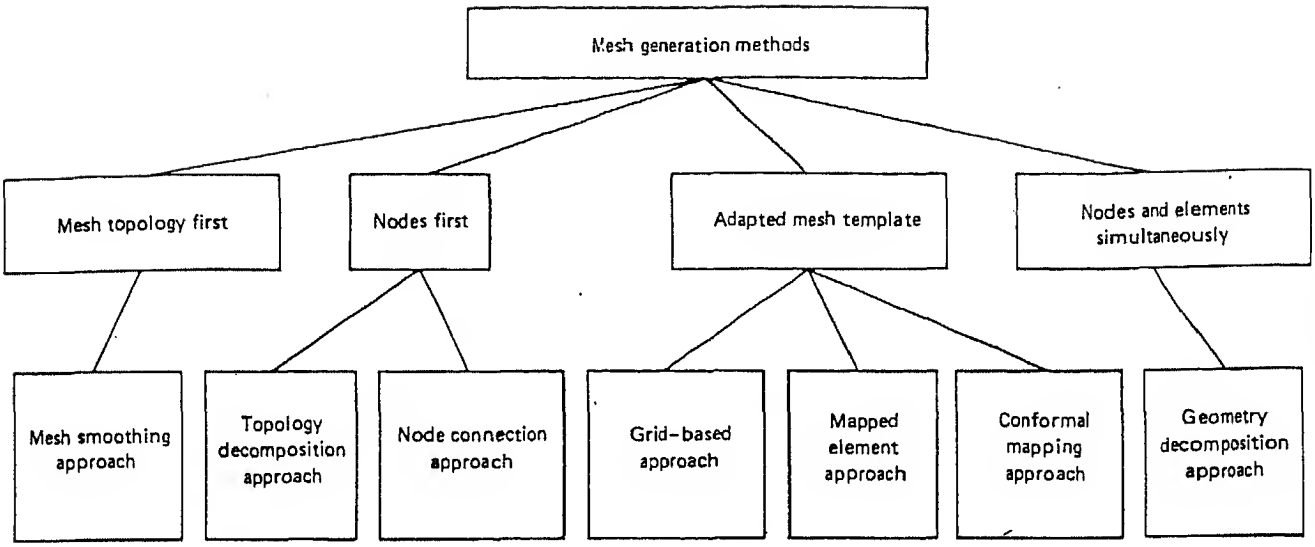


Figure Classification scheme for mesh generation methods

Table 1. Comparison of the mesh generation approaches

Approach	Quadri-lateral	Brick	Element shape	Mesh density control	Time efficiency
Topological decomposition	No	No	Poor	No	$O(N^2)$
Node connection	Yes	No	2D good, 3D fair	Yes	$O(N)$
Grid-based	Yes	Relatively easy	Interior elements excellent	Yes	$O(N)$
Geometric decomposition	Yes	No	2D good, 3D unknown	Yes	unknown

(N = number of nodes in the mesh)

Current serial mesh generating schemes can be classified as:

1. Interpolation mesh generation
2. Automatic triangulation
3. Quadtree/Octree approach
4. Mesh generation based on Constructive Solid Geometry

The interpolation mesh generation approach [CAV74]partitions the structure into simpler subregions first and then meshes each subregion individually. Though this technique is quite popular because of its ability to produce well conditioned meshes. It is relatively difficult to carry out local mesh refinements because of the need to fit together in a consistent manner,the individual meshes defined by each subregion of the structure.

Automatic triangulation [SIB78] has been popular method used in 2D mesh generation. However, this technique occasionally produces ill conditioned meshes and also it requires the extensive checking to ensure that newly formed elements do not pierce through or intersect any already defined ones.

The third approach is based on *Quadtree* encoding of the object in 2D and *Octree* encoding in 3D extension.[YSH83] The object is enclosed in a square universe which is then recursively subdivided into quadrants. The quadrants that are inside the object are retained and contribute to the elements of the mesh, those outside are discarded and those intersecting with the boundary are further subdivided. The subdivision continues till a prescribed level of the subdivision is reached. The quadrants at

the last level of the subdivision and intersecting the boundary are cut with respect to boundary and selective actions are taken to ensure that the result is a valid set of triangular elements. All the *in* quadrants are also split to ensure a valid mesh.

Mesh generation based on *CSG* (Constructive Solid Geometry) tree, generates the meshes for the object which can be represented by a CSG tree.[LEE83] It uses CSG tree to construct a set of *well distributed* points within the space of CSG objects and then uses a decision making process to construct a mesh of a this set of points. Point membership classification is extremely useful in point generation algorithms. The efficiency of mesh construction depends upon the size of the decision tree. This approach can generate a good mesh for a CSG object with constant mesh density throughout the object. However, extension of this approach to variable mesh density needs special care.

2.3 3D MESH GENERATION

Standard element type in 3D Finite element method includes *hexagon* (brick), *pentahedron*(wedge) and *tetrahedron*. These building blocks must be assembled in a consistent way , i.e. *vertex to vertex* fashion. It is extremely difficult to decompose a solid into valid i.e. disjoint, compatible and consistent union of solid finite element elements. The mesh generation problem becomes especially difficult when a variation of element density from region to region is required. Local mesh refinement is often a rule than the exception.[CAV85]

The three dimensinal finite element mesh generation techniques can also be classified as

1. Interpolators
2. Automatic Triangulation
3. Semiautomatic Point Insertion Method
4. Octree Modeling Technoique

The most successful technique is called as isoparamtric mesh generation. In this method an isoparametric matching is constructed from the conical domain (e.g. a unit cube) on a block structure. Then this mapping includces a natural coordinatisation on the block with respect to which elemental node types can be referenced. This approach suffers from the serious problem of grading. [ZIW83]

In the second approach, which is simmilar to 2D triangulation, the strategy of element generation is to select an edge connecting two node points in the solid and then surround this edge with tetrahedron defined by nearby points. During generation extensive checks must be made to ensure newly formed elements do not pierce through the early defined ones. Automatic triangulation is the same as that of 2D elements , the only difference is the elements are tetrahedron shaped.[CAV85]

The octree modeling technique is along the parallel lines to the modified quadtree technique.

2.4 Parallel Mesh Generators

Although there is little disagreement for the advantages of the parallel methods few points should be considered before parallelisation of any method. Speed is the main advantage of the

parallel architectures, but all algorithms can not be efficiently parallelised. Parallelisation is essential for the speed and it is important to explore the new areas of applications where parallel machines can be used. The method used here is an attempt to explore such a possibility.

E. Marks[MAR88] gives the parallel algorithm for the triangulation of the given points. He gives the optimal parallel algorithm for triangulating a set of points in the plane. Given n points in the plane, join them by non intersecting line segments such that every region interior to convex hull is a triangle. The algorithm is of $o(log\ n)$ time using $o(n)$ processors and $o(n)$ space.

F. Chang et al.[CHE89], have proposed a new scheme which uses a vertex label assignments scheme to provide the information for the image generation so that the parallel mesh generation on the basis of individual faces becomes possible. No checking of conformity is required as it is automatically assured. It carries the process of mesh generation by controlling the labels assigned to vertices rather than subdivision levels assigned to faces of regular quadrilateral mesh. The subdivision process, which can be of two types, that is balanced or unbalanced is entirely driven by the values of labels assigned to the vertices of the input network the algorithm performs subdivision of all the faces simultaneously, each by one processor. Each face of the network represents the root of the quadtree and is subdivided and labels are also assigned to the vertices of these subquads. This process is continued till all the labels are equal to zero.

The parallel modified approach , which we have used is very attractive alternative for the parallelisation of the mesh generation method. We can see that very few algorithms are inherent nature of parallelisation. Quadtree is one such data structure which can be used for such purpose. Generally the communication overhead amongst the nodes decries the advantages of parallelisation. But in our case as each quad can be handled independently, minimum communication is required in between the nodes or server and worker model. In this method , each quadrant is handled independently by each processor.Collecting the node information is only done by the server.The main drawback of the parallel programs is that they are highly architecture dependant. The parallel model of transputers is the attractive alternative for the quadtree generation which is explained in chapter 4 in details.The basic modified quadtree technique is explained below.

2.5 MODIFIED QUADTREE METHOD OF MESH GENERATION

The *Modified Quadtree* method uses, conventional quadtree encoding technique with some modification for the triangularisation technique.[YSH83] Mesh grading is proper in this case, in such a way that, a finer mesh in the regions of expected steeper gradient of field variables and a coarser mesh in the other places is possible.

The Quadtree approach seems attractive because of its inherent nature of parallelisation. As faster computers are available, so automating the preprocessor for FEM which is virtually the whole

time consumer and a tedious task, this method seems interesting.

The Quadtree mesh generation schemes goes this way:

In conventional coding of the quadtree, the object of interest is placed inside the square, that already has an integer coordinate system defined within it to identify the squares used in the quadtree representation. This square is subdivided into four quadrants. Next each quadrant is tested to see if it is inside the object(*BLACK*), or completely outside the object (*WHITE*) or only partially in (*GRAY*) . Homogeneous quadrants can be flagged as black or white , but the gray quadrants are again subdivided into four subquadrants. These quadrants are tested in the same manner as above until some prescribed resolution level is reached. The advantages of the object representation by this way are, the storage requirement is far less than the pixel representation and also the boolean operations, translational, rotational operations are faster.

But the object represented in this traditional way has the drawbacks also. They are :

1. In some cases the interior of the object may be represented by very large elements than the boundary elements.
2. Some neighbouring quadrants may undergo more subdivision than the others.
3. Since the object is represented as the set of squares , non vertical and non horizontal boundaries had to be expressed in terms of the squares. So there is a chance of loosing some of the information while coding in this way.

While encoding the object non horizontal and non vertical boundaries undergo repeated division engrossing the memory requirements.

These problems can be solved by modifying the quadtree encoding technique. Quadtrees when modified use the less space which is the main objective. In this technique, there is one more type of quadrant defined as the *cut quadrant*. This quad can have corners cut in any fashion. Thus when the required resolution is achieved, instead of representing it by *black* nodes, it is stored as a *cut* quadrant, with the intersection points information. In this way, we can represent the cut quad for any angular cut and there is no need of the extensive checking and the readjustments of the cut points to boundary points as proposed by Yerry and Shephard's scheme.[YSH83]

In this way we can not have entirely integer tree , and hence there are no problems such as storage of the array and retrieving the information. Because in any case the floating point information has to be stored.

The key to information stored at the node level are the discrete intersection of entities defining the boundary of the object with the boundary features of the node. The mesh is generated leaf by leaf method. The tree structure is used to transfer cell face by triangular information to neighbours to ensure the matching of the meshes within each leaf. The tree structure is also critical for containing the distribution of elements throughout the domain. The current implementation of the

procedure requires the neighbouring leaf nodes differ in size by only one level. This ensures the generation of smooth mesh gradation. Since the element size is dictated by leaf size, any spatially based procedures that can dictate the size of cells, can be used to control the distribution of the elements throughout the mesh.

Use of an optimal mesh should produce a solution of satisfactory accuracy for less computational cost. Unfortunately optimal meshes are extremely difficult to construct for most of the cases. So in this case we can start with a preliminary solution of coarser grid and develop procedures for automatically improving it by introducing finer meshes in the region where accuracy is inadequate.

Just construction of Modified Quadtree do not produce satisfactory element mesh since the quadtree/octree approach itself is not an element generation technique, rather it serves to simplify the geometry for an associated element generation scheme. The rich data structure provides a high degree of local control over the meshing process. So additional features must be added to create transitions between quadrants of different levels to ensure reasonable aspect ratio and obtain best possible mesh. To produce a valid mesh in which no common nodes lie along with the edge of the another element, transition meshes must be replaced between the quadrants of different levels. But in order to limit the number of multilevel transition meshes, a second modification is carried out on the modified quadtree to ensure only one level difference between any two neighbouring quadrants.

Afterwards the internal and boundary nodes can be triangulated independently to give valid mesh. The element are improved by the application of *Laplacian* operation. This operation places the interior nodes at the centroid of the nodes that they are connected.

The finite quadtree and octree mesh generation procedures are well suited for the adaptive mesh refinements also. Starting with an initial coarse mesh the nodes which are having large gradients can be bisected independently and the new mesh is generated. Computational growth has been used as a formal indicator of the performance of mesh generations schemes. THis indicator measures the rate of increase of CPU time with increasing number of elements. The advantage of this measure over the use of just CPU tIme alone is, it is independent of the computational speed of the machine used. In general a plot of CPU time vs number of elements is used as a measure of growth rate of mesh generation scheme. The best growth rate upto this time is found in quadtree schemes in which the CPU time is linearly proportional to the number of elements generated.
quadtree mesh generation scheme localizes elements formation to within the individual quadrants. When the information must be exchanged between a terminal boundary quadrant and its neighbours there are only four directions in which to conduct a search, no matter the level of the quadrant. More over the interior quadrants can be directly triangulated with out any further processing. This results in a linear rate for the quadtree mesh generation scheme.

2.6 MESH SMOOTHING

After the triangulation process the mesh smoothing techniques are applied. These are repetitive processes which reduce the overall speed of the execution. Often the elements produced by an automatic mesh generator are not well shaped. In such cases it is necessary to apply the mesh smoothing techniques to improve the mesh. The most popular technique is Laplacian technique which seeks the repcsitioning of of the nodes such that each internal node is at the centroid of the polygon formed by the its connected neighbours. The repositioning is usually done iteratively.

Automatic mesh generation has still number of disadvantages. No CAD system can supports necessary handles for the mesh generation which will allow material properties and load cases to be specified with the geometric model. All automatic mesh generators are therefore, forced to create more dense meshes than would be otherwise be necessary.In order to replace the mesh generation schemes by more reliable scientific methods and total automation, a more precise knowledge of convergence and absolute errors of results are necessary.

The state of the art in mesh generation is such that fully automatic mesh generation with adaptive capabilities is not yet fully a reality. The requirement of mesh density is based on the observation that the use of less dense mesh either locally or globally after rediscretization can lead to poor results with the same problem solved with remeshing.

CHAPTER 3

QUADTREES AND OCTREES

3.1 Introduction to quadtrees and octrees.
3.2 Parallel implementation of quadtrees.
3.3 Octrees.
3.4 Algorithms for Quadtrees.

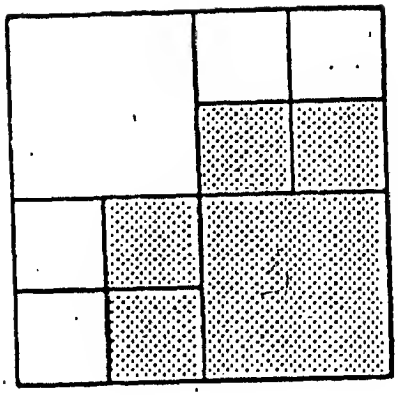
3.1 Quadtrees and Octrees

There are various data structures to represent image data. There are numerous hierarchical data structuring techniques in use for representing spatial data. Hierarchical data structures are important techniques in the areas of image processing, computer graphics, solid modeling, geographic information systems etc..

They are based on the principle of recursive subdivision. They are mainly useful because of their ability to focus on the interesting subset of the data. The hierarchical organization of the image space allows the resolution to vary with the complexity of the objects in various regions. They are used as a basic geometric modeling data structures. Many operations on the images can be performed with other types of data structures, however, hierarchical data structures are attractive because of their conceptual clarity, ease of implementation and inherent

characteristic for parallalization. There are various types of hierarchical data structures as pyramids, quadtrees, octrees, polytrees. The term *Quadtree* is used to describe a class of hierarchical data structures whose common property is that they are based on the recursive subdivision of the continuum. It is a type of data structure used for compact representation of 2D images. It is based on the recursive subdivision of the space. Quadtrees are same as the normal binary trees but with degree four. Generally a node is either a leaf or having the four sons. A quadtree is generated by dividing the image with quadrants and repeatedly subdividing the quadrants into sub_quadrants until each quadrant has uniform color. The stopping rule of the formation of the quadtree is a leaf criteria. The modification of leaf criteria can be the depth of the tree. Once the maximum depth is reached , even if the leaf criteria is not satisfied the region is simply represented as a leaf node.

But in such a case it is possible to loose some of the information. The leaf criteria may be that the quad may lie completely inside the object, represented as *BLACK* node or leaf, or it may be completely outside the object, represented as *WHITE* node, or it may be cut at the boundaries of the object, represented as *GRAY* node, for which again subdivision is necessary. Thus heterogeneous node has four descendants called as *SW, SE, NW, NE* corresponding to the southwest, southeast, northwest and northeast quadrants of the parental quad Thus a node at level h corresponds to 2^{k-h} sized quad, where 2^n is the size of the original image space. This can be modified to the maximum depth technique. In this technique ,even if the leaf criteria is satisfied, subdivision takes place of completely inside nodes till the maximum prescribed level of the tree formation is not reached.



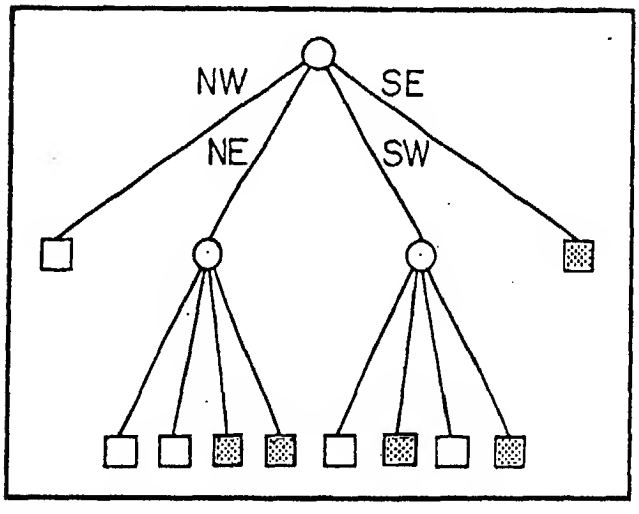


Figure . Pointer encoding of the quadtree of Figure . . Internal nodes are represented by circular nodes. Terminal nodes are represented by square nodes whose contents correspond to the blocks in Figure

The quadtrees can be differentiated on the following basis

1. The type of data it represents such as region, point, curve , surface or volume.
2. The principle of decomposition process such as regular or irregular quadtrees. In case of regular quadtrees decomposition at each level is into the equal parts such as hexagonal , square quadtrees [SAM88]

Regular decomposition type of quadtrees are easy to formulate and program and not much heuristic is involved. In case of irregular division, the division is governed by the nature of the input data. Even though there can be compact representation, too much heuristic is involved in this. There are various ways to store the quadtrees, such as classical quadtrees, linear quadtrees. The amount of storage required by the quadtrees or octrees is directly proportional to the number of leaf nodes. Generally a node is represented by a record with pointers to the four children. There is also a field for the identification of the type of node such as *gray, black or white*. If the node is a leaf node then all pointers to son are NULL. The main limitation of quadtrees is that the the normal classical structure makes it spacewise expensive as the leaf nodes have the null children. So various other ways of storing the quadtrees has been explored. Linear quadtrees is one of the modification for the representation of quadtrees. In this method, the pointerless representation of the quadtrees has been implemented. But classical quadtrees are conceptually clear and traversal algorithms are very easy and faster. Most of the quadtree algorithms are simply preorder traversal and hence their execution time is a linear function of number of nodes in the quadtree. The root of the quadtree corresponds to the image it represents. A node in a quadtree is

either a leaf or has four node son i.e. non terminal node. Each son node is associated with a quadrant of the block corresponding to the father node and the original image. [SAM84]

The advantage of the quadtree representation for images is that simple and well developed tree traversal algorithms allow faster execution of certain operations such as superposition of image area and perimeter calculation. In addition the 2D coordinate of each block is implicitly stored in and can be readily recovered from the quadtree representation. So there is no need of explicitly storing the information of the co-ordinates of the points of the quad for which the node is represented. Each node of a quadtree represents a definite corresponding block in the original image. Given a polygon with v vertices , construction of quadtree within the space of $2 * 2$ has execution time of $O(v*n)$. One of the most useful features of the quadtree is that an increase of the resolution parameter requires further subdivision of the terminal nodes only, leaving the remaining part of the quadtree unchanged. Roughly speaking a refinement of the image merely requires the refinement of the leaf only. Thus quadtrees are dynamic structures.

3.2 Parallel implementation of the quadtrees

There are various parallel quadtree building algorithms. Dyer has described a bottom up algorithm which builds a quadtree by recursively merging the blocks of the same color starting from the pixel level.[DYR81]. Eiedeman has presented a Concurrent prolog algorithm for quadtree building.[EDR85] Another approach is using shuffled row major indexing scheme for the pixels in the given image. Here, for each pixel it is determined, whether it can

represent a leaf node and if it can ,the size of the leaf node is determined by the number of its preceding pixels and the number of its succeeding pixels in the run it belongs to. The trend in the parallel algorithms to construct the quadtrees is mostly from the image processing point of view. These algorithms are basically meant for the data which generally comes as a raster or pixel format. They are mostly bottom up approaches in which the tree is formed by merging the same coloured neighbouring pixels to form the leaf. The approach generally changes depending upon the format in which data comes and how it is arranged. There are not many explorations such as the top to bottom approach and any other ways.[88]

3.3 Octrees

The modeling of 3D objects and space by computer is important in space planning , computer animation and a machine vision. An extension of the quadtree structure is the octree structure for the representation of the 3D objects. The octree generation is done on the same basis as the quadtrees. An octree is generated by dividing a universe into octants and sub-dividing the octants into sub-octants with all voxels in each octant lie entirely within or outside of an object. Thus octree nodes which are called as voxels represent the cubic volume in the original image. The voxel is either a leaf or internal voxel. Internal voxels have the eight children and the leaf nodes do not have children at all.

The octree advantages over other methods for solid modeling are: Any arbitrarily shaped objects convex, concave, with interior holes can be represented in the required precision. Only

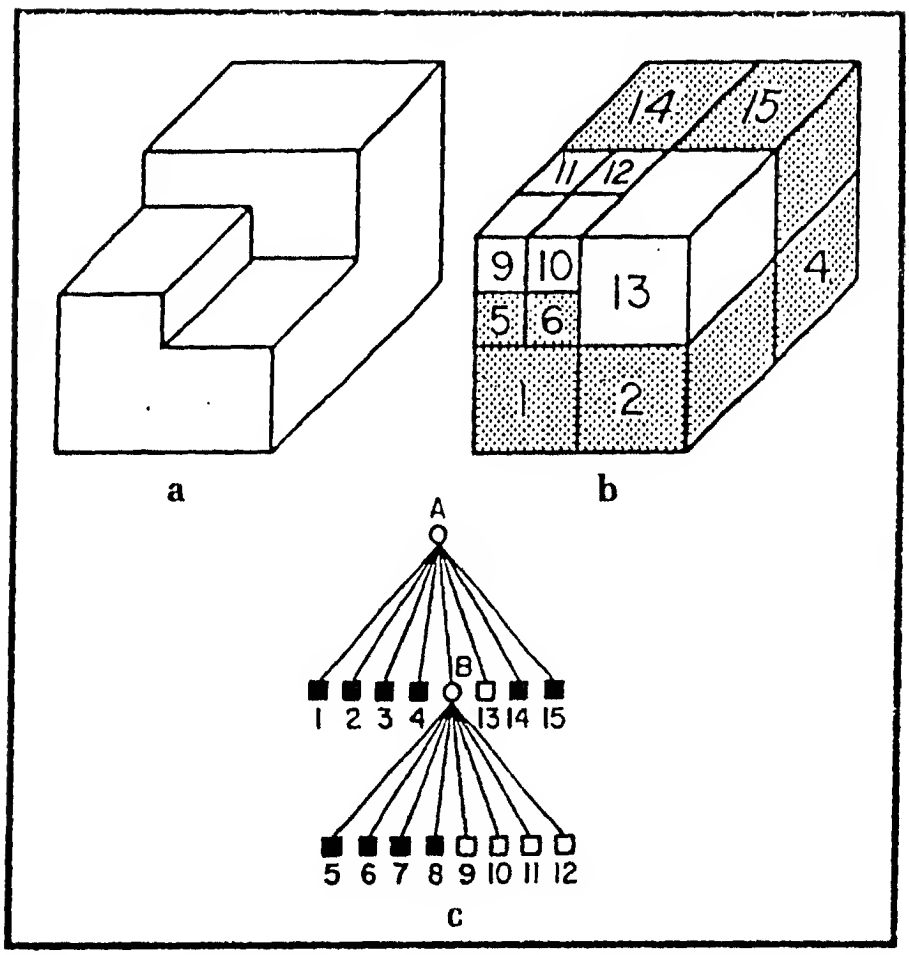


Figure . (a) Example 3D object, (b) its octree block decomposition, and (c) its tree representation.

a single set of manipulation and analysis algorithms are required for all objects. New techniques are not needed to handle sophisticated or complex shapes. Because of spatial sorting and uniformity, hidden surface display algorithms requires no searching and sorting. The main advantage of the octree model is the linear complexity of the boolean operations and rendering algorithms. Several geometric properties such as surface area, interference are easily calculated. However they are approximate models as the slanted surfaces must be represented by the minimum scale black nodes. The modified quadtree/octree approach reduces the degree of sub_division and therefore requires less storage than the classical octrees.

The main disadvantage is the large memory requirement for encoding. The algorithms generally are *O(surface area of the object)* for the volumetric data and for quadtrees generally *O (perimetere of the object)*

There is a large literature available on various operations on the quadtrees such as

1. Algorithms for the quadtree formation.
2. various operations such as transformation or rotation of the quadtree.
3. Neighbour finding.
4. From raster image point of view.

Out of that from this work point of view very few algorithms are useful. The one technique we have used is explained in detail over here.

3.4 Neighbour finding algorithms

In several applications, the computation of each node of the quadtree is dependant on the value of its adjacent neighbours. Thus we must be able to locate the neighbour of a node.

Two tiles are neighbours if they are adjacent either along an edge or vertex. The requirement for our algorithm was to find the neighbour along the edge and not along the corners. The neighbour finding operation must performed in a most general way i.e. the neighbours must be located in a way that is independent of both size and the position of the node. Locating adjascent neighbours in the horizontal or vertical direction is relatively straightforward. The basic idea is to ascend the quadtree until a common ancestor with the neighbour is located. And then descend back down the quadtree in search of the neighbouring node. Neighbours along the edges need not correspond to the block of the same size. If the neighbour is larger then only part of the path from the nearest common ancestor is retraced, otherwise the neighbour is of the same size of the node. If there is no neighbour as in the case of border nodes then NULL is returned. The nearest ancestor which is common is the first ancestor node which is reached via its sons. We retrace the path used to locate the nearest common ancestor except that we make a mirror image moves about an axis formed by the common boundary between the nodes. For example in the case of W neighbor the mirror images of NW and SW are NE and SE.

CHAPTER 4

SOFTWARE DEVELOPMENT

4.1 INTRODUCTION

Here in this chapter , actual implementation details are given Also given is how the modified quadtree method used for this purpose is different from the original method proposed by Yerry and Shephard's idea of integer tree. The details of the parallel architecture used are also provided. The parallel algorithm for mesh generation, implementation problems, the limitation of the architecture for the method to parallelise are also discussed. The next chapter gives the detail information of the results, Comparison between serial and parallel methods and the scope for the future work.

The actual implementation of the algorithm can be divided into five modules :-

1. Data Acquisition.
2. Quadtree Generation
3. Valid Quadtree Formation.
4. Actual Triangulation.
5. Global Node Numbering and Connectivity.

The only input required for the modified quadtree technique is the boundary points. The method assures the valid mesh in most of the cases, as the finer elements in the area of steeper gradient of variables and coarser elements elsewhere. So the minimum mesh refinement is required. But sometimes it is difficult to shift the newly created points on the boundary after formation of the mesh. The drawback is particularly true for current method because of its spatial sensitivity. So sometimes required number of points on the boundary are also specified in order to get more refined mesh. Although we have assumed only straight line polygons as the input for the present work, it is easy to write a preprocessor which can accept any shaped geometrical object input, from any standard CAD package and is suitably converted into its polygonal representation.

4.2 Quadtree Generation

Once the input polygonal data, in which polygons can have any shape, either singly or multiply connected, is acquired, it is enclosed by the square of size $2^n * 2^n$ n depends upon the co-ordinate information of the polygon. As already explained the quadtree encoding goes like this as :

The original square is divided into four equal subquadrants. Now each quadrant is checked against polygon whether it completely inside the object (*BLACK*), or completely outside the object(*WHITE*), or partially cut by the polygon boundaries.(*GRAY*) The *GRAY* quadrants are again subdivided in the same way. The quadtree formation is controlled by two parameters , *Max_depth* and *Min-Resolution*. Max_depth gives the maximum depth of the tree upto which the tree formation should go, irrespective of its status. Min_resolution defines the stopping criteria for the leaf. In generating quadtree even if the quad is completely inside the object , it is subdivided into four BLACK nodes if the required depth is not yet reached. The gray quadrants are termed as *gray* or *cut* according to the depth of the tree that has been reached. Instead of passing the original polygonal structure everytime to each quad, only the parental cut quad polygon information is passed to its children. As the determination of the status of the quad is a function of number of edges of the of the polygon, significant speedup is achieved. Normal classical way of pointer tree has been used to represent the quadtree structure mainly for two reasons. One, as the ease of implementation and maintainance and also it is conceptually very simple. Second the efficient neighbour finding algorithm explicitly needs the quadtree structure in the classical tree form. With other types of representations explicit pointers to nighbouring nodes are required, which kills the very purpose of using the other type representations. The main constraint to the classical quadtree representation is the pointer overhead. The parallel transputer model which we have used limits no way this representation as the individual nodes have sufficient memory. (4Mb RAM for each node).

The individual status of the quad is determined against each edge of the polygon. We have used *Weiler_Athertron* algorithm to determine the cut surface of the polygon against the quadrant. [WEI77] Although it is somewhat complicated to implement, the popular *Sutherland Hodgeman* algorithm is not used as it gives a lot of dangling regenerative edge cases for the polygons with the holes which are difficult to eliminate. [SUT74]

The main difference between Yerry-Shephards modified quadtree algorithm and the present work is that there is no concept of integer tree involved. The integer tree is unnecessary in the sense that final point representation should be, anyhow in terms of the floating points. So all basic purpose of using integer tree is lost. In the above method cut quadrants are represented as the integer points, in which the quad is subdivided into definite intervals. To determine the nature of the cut quad, almost eighty enumerated cases [GAN88] have to be explicitly checked. Also we have to store this information in a stack which requires another complicated algorithm to retrieve the information. Due to this also, after forming the mesh, there is a requirement to pull back the node points to the boundary and to be stored as a proper floating point representation. All this can be avoided with a simple memory overhead of one pointer for each leaf, which is used to represent the resultant cut quad poly after cutting the polygon by the quadrant. There is no need to check the cases, retrieving the information and pulling back the boundary nodes. Also there is no explicit need to determine the boundary nodes as a complex boundary discrimination algorithm is used in the above mentioned case.

4.3 Validation of the Quadtree

Once the quadtree is formed , it can not be used directly for the mesh generation. Additional features must be added and checked to create transition between quadrants of different levels. To produce a valid element mesh in which no corner nodes lie along the edge of another element, transition meshes must be placed between quadrants of different levels. Such a situation allows gaps to open in the structure, unless the continuity of the displacement is ensured. So in order to limit the number of multilevel transition meshes, a modification of tree structure is carried out which ensures only a one level difference between any two neighbouring quadtrants in all four directions. In this case, the conformity of mesh should also be assured to avoid gaps in the structure, i.e. the intersection of two non disjoint, non identical elements consists of a common vertex or edge. The algorithm explained in the last section is used to find out the neighbours of each leaf in the four directions. If the level difference is found to be more than 1 then the particular leaf is individually subdivided without modifying the rest of the quadtree structure. Thus after checking the valid level difference between each node and all its neighbours as less than or equal to one, the tree is ready for direct triangularisation.

4.4 Triangularisation, Node Numbering and Connectivity

In this work only triangular elements are implemented since these are the most flexible elements. The individual leaf nodes triangularisation is carried according to the geometry it represents. The square nodes are represented by the diagonal elements, triangular nodes are accepted without any modification.

Only the nodes with cut status and the nodes whose neighbour is at one level lower has to be carefully triangulated.

Once the triangularisation pattern is decided , global numbering is given to the each point and global connectivity information is stored. Finite element also requires the connectivity numbering of the nodes in the specific fashion so that the connecting points should have minimum difference between their numberings. How to do this is itself an independent problem, for which, in this work has not been given enough attention. But a postprocessor can be written to ensure the minimum width stiff matrices. The elements aspect ratio can not be ensured in the elements produced in this way so these are improved by the application of Laplacian operation. In this operation the position vectors *Pi* of the interior nodes are positioned to satisfy the expression $P_j = 1/n_i * \Sigma P_j \quad i=1, m$ where n_i is the number of nodes connected to the node i, P_j is the position vector of the two connected nodes and m is the total number of node points. This equation places the node at the centroid of the the nodes that they are connected to.

4.5 Parallel implementation of Algorithm

Although the advancement of technology has made the parallel architectures widely available, there are certain serious drawbacks generally faced by the parallel programmers. Existing parallel languages are tied to some architectural class. Because of paradigms and pattern of various parallel architectures differ, the programmer must be aware of the kind of architecture on which the software is going to be run, the number of processors, storage capacity and their configuration. So

before representing the actual parallel algorithm the parallel architecture is briefly explained here and afterwards the actual parallel algorithm is provided.

Parallel architectures can be broadly classified as

1. shared memory models
2. Message passing models.

In case of shared memory models the common memory is shared between all of its nodes and the issues like exclusive read/write operation comes into picture. In the other model of message passing there is no common memory between individual nodes. Each node has its own separate memory area, the communication is strictly through the message passing over the channels. Each model has their merits and demerits. The parallel architecture, which has been used is, *transputer* based. It is based on the message passing architectural model. There are four *TS800* nodes connected in a star fashion. Each node is having its own memory of 4Mb RAM. There is a one node which is connected to host PC and is a master to which all the other nodes of the network are connected. The work done is in *PAR_C*.

The treatment of parallel processing in transputer systems is based on the idea of *communicating sequential processes*.[HOR78] In this model, a computing system is a collection of concurrently active sequential processes which can only communicate with each other over channel. A *channel* is nothing but the one sided communication way which connects exactly one process to another process. Each process can have any number of output or input channels but they can not be dynamically created. The channel communication is synchronised. A process wanting to send the message over a channel is always forced to wait until the

receiving process reads the message. A process in this model is just a black box connected to the outside world only by its *channels*, thus the actual nature or implementation of the process is not important. Each transputer processor has four *Inmos* links to connect it with other transputers. Each link has two channels, one in each direction. These channels behave exactly like software channels and provide synchronised , unidirectional communication. Each processor can not be connected more than the four other processors. It is also possible to run any number of concurrent processes to be run on individual processor. The equivalence of internal channels means it is possible to develop a complete parallel system on a single transputer and then move some of its processes onto another processors without having to recompile any code.

Parallel C is based on the same above described model of transputers. A complete application is viewed as a collection of one or more concurrently executing tasks. Each task has its own memory for code and data, a vector of input ports and output ports. The code of a task is a single transputer image file generated by a linker. Each element in a input and output vectors is of type *pointer to channel word (CHAN *)* Ports are bound to real channel addresses by configuration software external to the task. This software also provides ways of specifying which software tasks are to be run on which hardware processors.

For many parallel computations it is useful to be able to create applications which will automatically configure themselves to run on any network of transputers. Such applications will run automatically faster when more transputers are added to the network without reconfiguration or recompilation. This technique

is called as *processor farming*. In the processor farm technique, an application is coded as one master task which breaks the job down to small independent work packets which are processed by any number of anonymous worker tasks.All worker tasks must run the same code. Each worker simply accepts the packet over the network processes it and sends the result packet back to the master via routing software.

The quadtree method is attractive from parallelisation point of view. The architecture we have used is suitable for this. The model consists of a root node which is connected to the host of the PC. The other three node transputers are directly connected to the root. The communication between individual nodes is unnecessary for this application, since most of the information processing is done individually. Generally speedup achieved by the parallelisation is marred by the heavy communication requirements between the neighbouring nodes and the expected speedup is not obtained, but in this case there is a little communication overhead. In our case the communication overhead was minimum since we can build the quadtree truly independent of each quads. Only communication to master is required.

The five modules which should be carried sequentially, the parallelisation is done according to the individual model.

1. Data acquisition is done by the root node.
2. The two tasks are there. One task which determines the status of the quad and the resultant cut polygon for the node. Other task is a master task which waits for the four workers to complete the work and takes the decision of the further subdivision of the quad.The worker task is kept on the four nodes. On the root the two task

run concurrently.

Thus serial tree formation is a function of total number of nodes in the tree , while parallel version is a *f*(total no of GRAY nodes in the tree) so for the larger depth of the trees the speedup of almost the number of processors is achieved. Thus if *N* is the total number of nodes in the tree, The number of *Gray* nodes = (N-1)/4 . Since the quadtree node has either four children or none. Thus a speedup of the factor 4 is achieved.

The formation of the quadtree thus can be done parallelly. the validity of the tree has to be checked by the root node as it involves the information of the neighbouring node, which can not be done truly independently . The level difference between the node and its neighbour is checked parallely and the worker sends this information to the root node , where root node takes the proper decision of balancing of the tree and ensures the level difference of one in between all the leaf nodes and their respective neighbours.

The valid tree leaf nodes can individually triangulated according to the type of the polygon it cuts and this information is passed to the root node . Root node accepts the connectivity information and the co_ordinates of the nodes and gives the global numbering as well as stores the connectivity information for the mesh generator's final output.

1. Parallel Quadtree Generation

```
PAR_BUILD_TREE  (tree,  quad)
if  (tree->color  ==  'GRAY')  then
            PAR_DET_COLOR_OF_CHILDREN(tree, quad);
            subdivide quads into four children quads;
            PAR_BUILD_TREE(tree->ch[1],    quad1)
```

```
PAR_BUILD_TREE(tree->ch[2], quad2)
PAR_BUILD_TREE(tree->ch[3], quad3)
PAR_BUILD_TREE(tree->ch[4], quad4)
```

The procedure PAR_DET_COLOR_OF_CHILDREN is a task on each worker processor, altogether the four worker processes determine the status of the four son quads.

2.Validity of the tree

```
For leaf_nodes
      do
          assign leaf to each worker.
          GET_PAR_LEVEL_DIFF(leaf);
          if (level_diff  > 1)  then
                                PAR_DET_COLOR(leaf, quad);
```

3.Generation of the Nodal connectivity

assign each worker the leaf;
get the local nodal coordinates and the connectivity information.
give the global nodal numbering and the connectivity information.

The smoothing operation is handled by the root serially as the shared memory is not available. When the co_ordinate information is changed , the new information must be used by each worker. The model of message passing does not allow to do it efficiently as large communication overhead is involved.

The next section gives the actual experimental results for serial as well as parallel implementation . Also proves the results experimentally.

CHAPTER 5

CONCLUSIONS

The algorithm described in the preceeding chapters has been successfully implemented on TRS 800 transputer based PC system. It has been tested on several data sets. The algorithm gives satisfactory results for a number of geometries such as star shaped polygons, concave as well as singly or multiply connected polygon boundaries. There is almost no user intervention needed. The algorithm gives robust control over mesh generation using two parameters: i)the Max_depth ii) Min_resolution.

Despite the fact that the algorithm works with different kinds of data sets it suffers from certain limitations. The quadtree structure is highly sensitive to the spatial arrangements. It is possible that for the same geometric structure defined in different co-ordinate references one may getdifferent triangulations. This is not an algorithmic problem but implementational difficulty. The geometric problems are difficult to implement since the number of possible cases is inordinately large. very difficult to consider all the specialised cases.In formation of the quadtree if such unatterded cases arrived , then it is possible to get the erroneous results. For the present work such cases are avoided.

Although this implementation is complete in itself , some modifications are necessary.

- Present algorithm only works for the 2D geometries. But 3D implementation can be carried out on the same grounds. Conceptually the ideas presented are can easily be extended for the 3D work. As the effort gives satisfactory results of parallelisation , similar efforts for octrees will be fruitful.

- The observation of the current implementation shows that the boundary elements are not well shaped. Special Techniques can be used to ensure better shaped elements. Some heuristic techniques can be applied to the problem of proper node numbering and the connectivity problem , as already mentioned.

- Although satisfactory speedup can be achieved because of parallelisation, the speed of parallel implementation can further be improved by careful and precautionary measures.

REFERENCES

[AHU86] N.Ahuja 'Volume/Surface Octrees For the Representation of 3D Objects' Computer Vision, Graphics & Image Processing. Vol 36, pp. 100-113 1986

[BEG90] Bern M, Eppistein P, Gilbert J. 'Provbly Good Mesh Generators' 31st annual sym. on FOCS Oct, pp.231-241.1990

[BHA90] Bhatia R, Lawrence K 'Two Dimensional FEM Generation.' Computers & structures vol. 36 No.2 pp. 309-319 1990

[CHE89] Cheng Y et al. 'A Parallel Mesh Generator Algorithm Based on the Vertex Lable Assignment' Int J. Num. Meth. Engg. vol.28 pp. 1429-1448. 1989

[COO74] Cook W, 'Body Oriented Co-ordinate for Generating 3D Meshes' Int. j. Num. Meth. Engg. vol.8 , pp. 27-34 1974

[CAV85] Cavendish j, Field D and Frey W, 'An Approach to Automatic 3D Finite Elmont Mesh Generation' Int. J. Num. Meth. Engg. Vo.21, pp. 329-347. 1985

[DYR81] Dyer C, Rosenfiels A. 'Parallel Image Processing by Memory Augmaented Cellular Automata' IEEE PAMI-3 , pp. 21-41.1984

[EDL85] Eiedeman s, Shapiro E , 'Quadtrees in Concurrent Prolog' Proc. Int'l conf.Parallel Processing, pp. 544-51 1985

[GAN88] Ganesha k, 'Automated Mesh Generation for Finite Element Modelling' Dept. of Mech. engg. IITK. 1988

[HIL90] Hill F, 'Computer Graphics' Maxwell-Macmillan 1990

[HOR78] Hoare C. 'Communicating Sequential Processes' Comm.ACM pp. 666-677 1978

[HUN87] Hung y, Rosenfield A, ' Parallel Processing of Linear Quadtrees on a Mesh Connected Computers' Tech report CSC-TR1817 March 1987, Univ. of Maryland, college park.

[KHO88] K Ho_le 'Finite Element Mesh Generation Methods : a Review and Classification.' CAD vol.20 No.1 Jan 1988 pp. 27-38.

[LEE85] Lee y, 'Automatic Mesh Generation of Finite Element Meshes Based on CSG' ph.d. thesis, Univ. of Leeds U.K.83

[LO85] Lo , 'A New Mesh Generation Scheme For Arbritary Planar Domains' Int. J. Num. Meth. Engg. vol.21 , 1985, pp. 1403-1426.

[MEG82] Meager ' Geometric Modelling using Octree Encoding.' Computer Graphics & image processing. vol 19 1982 pp. 129_147

[ROG85] Rogers D. 'Procrdural Elements for Computer Graphics' McGraw Hill, 1985.

[SAM82] H. Samet 'Finding Neighbour and Naming ' Computer Graphics & Image processing. Jan 1982 pp. 35-47.

[SAM84] Samet H 'The Quadtree and Related Hierarchical Data Structures ' ACM Computing surveys, vol. 16, No,2 1984 pp. 187-260

[SW188] Samet H, Webber R 'Hierarchical Data Structure and Algorithms for Computer Graphics' Part I, IEEE CG&A, May 1988 pp. 48-68

[Sw288] Samet H, Webber R 'Hierarchical Data Structure and Algorithms for Computer Graphics' Part HI, IEEE CG&A, July 1988 pp. 59-75

[SAM89] Samet H, Spatial Data Structures. Addison_wiley, 1989.

[SUT74] Sutherland I, Hodgeman 'Reentrant Polygon Clipping' CACM vol.17, 1974, pp. 32-42

[YSH83] Yerry M, Shephard M 'A Modified Quadtree Approach to Finite Element Mesh Generation' IEEE CG&A Feb 1983 pp 39-46.

[YSH84] Yerry M., Shephard M 'Automatic 3D Mesh Generation by Modified Octree Technique' Int J. Num Meth. Engg. vol.20 No. 11, Nov 84 pp. 1965-1990.

[WEL77] Weiler k, Atherton P, 'Hidden Suraface Removal using Polygon Area Sorting' Computer Graphics,vol.11, 1977 pp 214-223

[WOR84] Wordenwebr B, 'Finite Element Mesh Generation' CAD vol. 116, No.5 Sept.1984 pp. 285-291

' Transputers & Occam Programming . '
IEEE software Jan 1988. pp.69-78

[3LC90] 'Parallel C Ref. manual' 3LC Ltd.

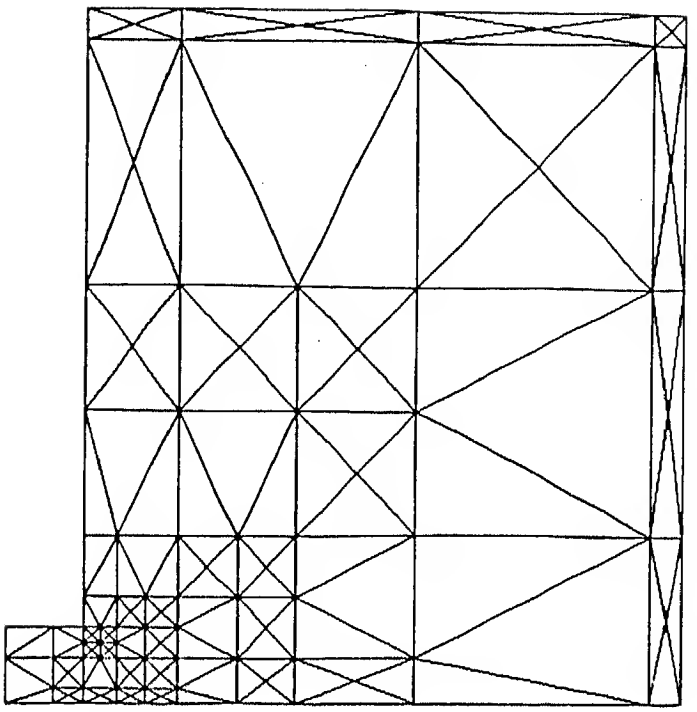
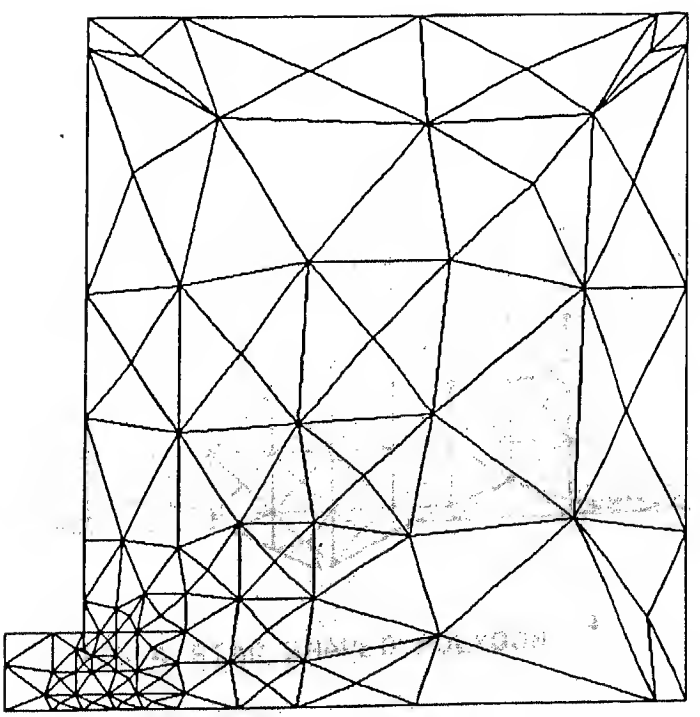


FIG.1 MESH AFTER FIRST PHASE



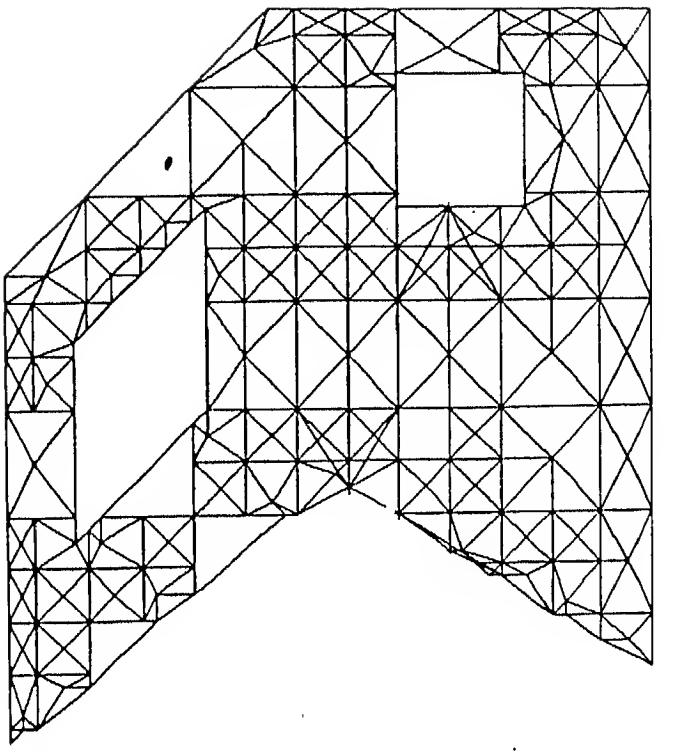


FIG.3 GEOMETRY WITH HOLES

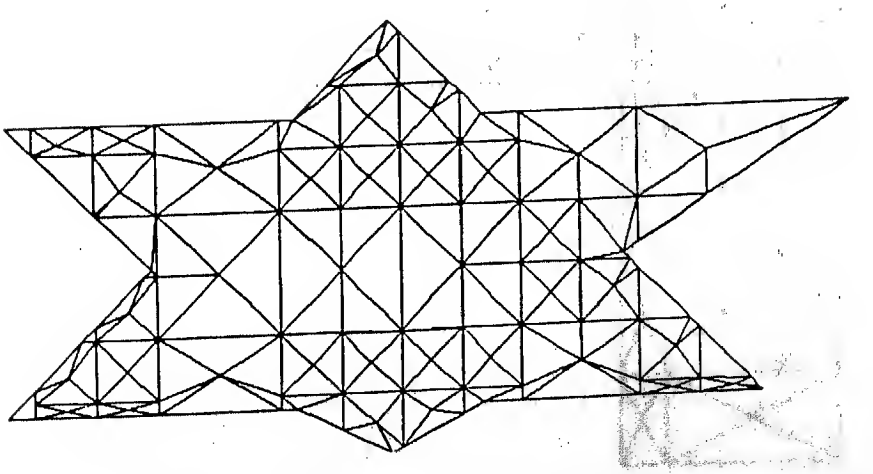


FIG.4 STAR SHAPED POLYGON

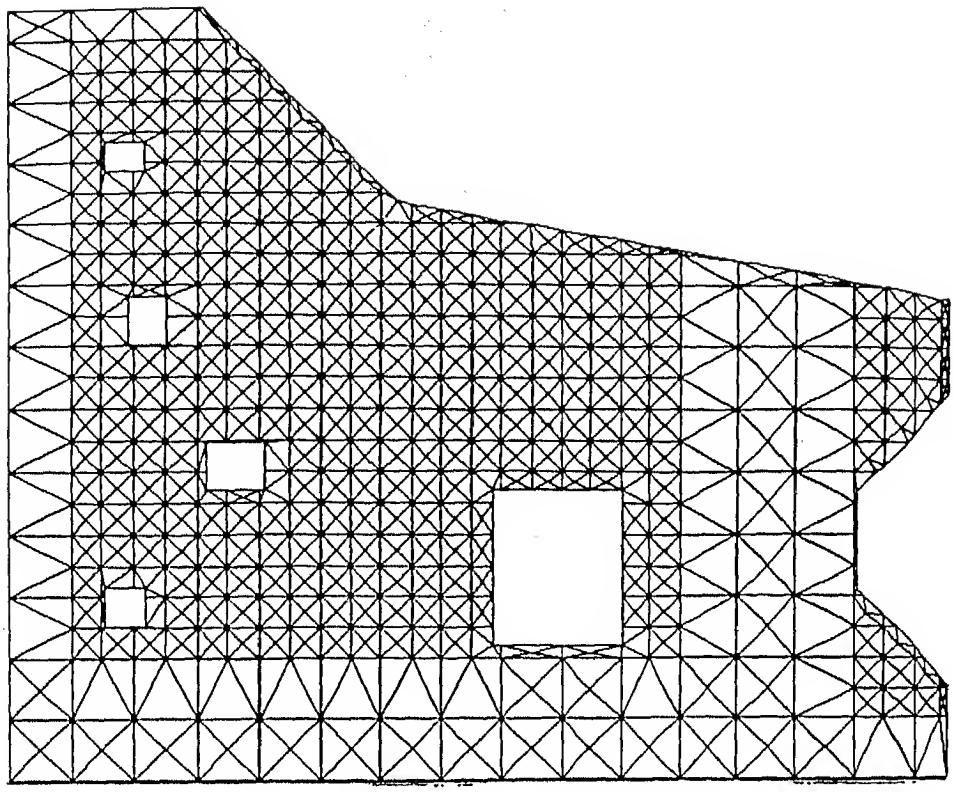


FIG.5 A VERY DENSE MESH

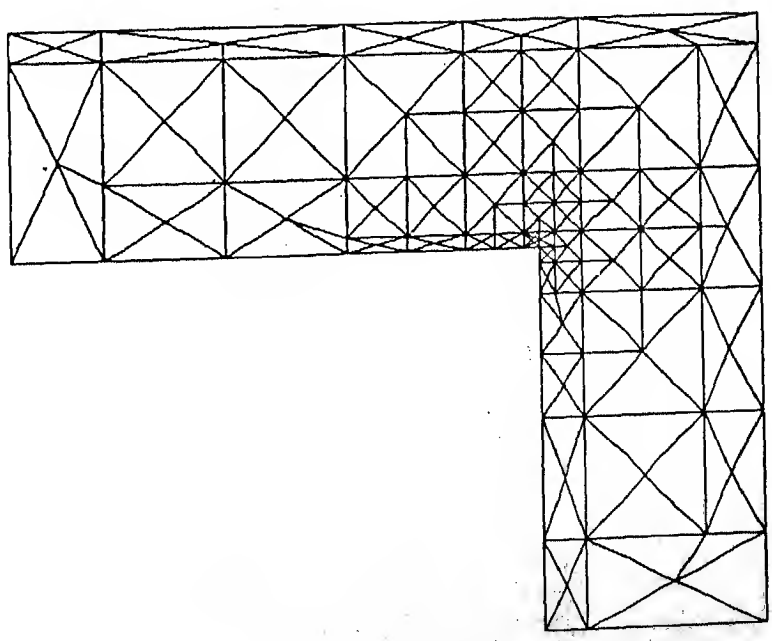


FIG.6

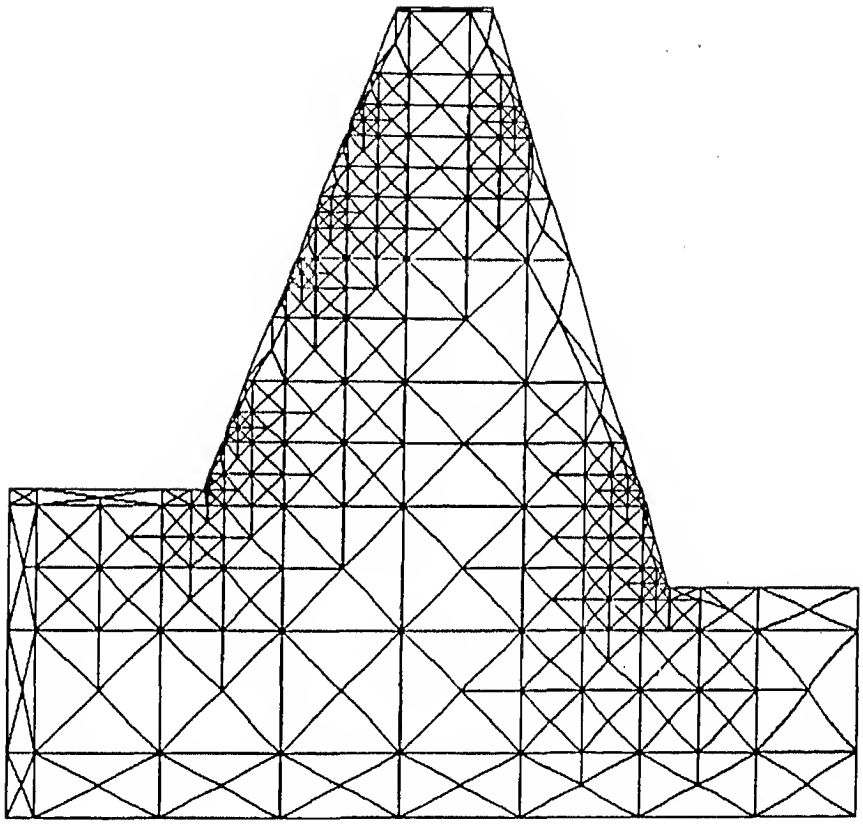


FIG.7

E-1991-M-ABH-DEV